Vorlesungsskript

Theoretische Informatik 2

Wintersemester 2009/10

Prof. Dr. Johannes Köbler
Humboldt-Universität zu Berlin
Lehrstuhl Komplexität und Kryptografie

22. Oktober 2009

Inhaltsverzeichnis

Inhaltsverzeichnis

1 Einleitung

In der Vorlesung ThI 1 standen die mathematischen Grundlagen der Informatik im Vordergrund. Insbesondere lernten Sie, wie man folgerichtig argumentiert und wie man formale Beweise führt. Als universelle Sprache der Mathematik lernten Sie dabei die mathematische Logik kennen, insbesondere die Aussagenlogik und darauf aufbauend die Prädikatenlogik. In dieser Sprache lassen sich nicht nur algebraische und relationale Strukturen modellieren, sondern auch Rechenmaschinen wie zum Beispiel die Turingmaschine.

Ein weiteres wichtiges Thema der VL ThI1 war die Frage, welche Probleme algorithmisch lösbar sind.

Themen der VL ThI1

- Mathem. Grundlagen der Informatik, Beweise führen, Modellierung (Aussagenlogik, Prädikatenlogik)
- Welche Probleme sind lösbar? (Berechenbarkeitstheorie)

Dagegen stehen in dieser Vorlesung folgende Fragen im Mittelpunkt.

Themen der VL ThI2

- Welche Rechenmodelle sind für bestimmte Aufgaben adäquat? (Automatentheorie)
- Welcher Aufwand ist zur Lösung eines algorithmischen Problems nötig? (Komplexitätstheorie)

Schließlich wird es in der VL ThI 3 in erster Linie um folgende Frage gehen.

Thema der VL ThI3

- Wie lassen sich eine Reihe von praktisch relevanten Problemstellungen möglichst effizient lösen? (Algorithmik)

Rechenmaschinen spielen in der Informatik eine zentrale Rolle. Hier beschäftigen wir uns mit mathematischen Modellen für Maschinentypen von unterschiedlicher Berechnungskraft. In der Vorlesung Theoretische Informatik 1 wurde die Turingmaschine als ein universales Berechnungsmodell eingeführt. In ThI3 wird das etwas flexibere Modell der Registermaschine (engl. random access machine; RAM) benutzt. Dieses Modell erlaubt den unmittelbaren Lese- und Schreibzugriff (random access) auf eine beliebige Speichereinheit (Register). Hier betrachten wir Einschränkungen des TM-Modells, die vielfältige praktische Anwendungen haben, wie z.B. endliche Automaten (DFA, NFA), Kellerautomaten (PDA, DPDA) etc.

Der Begriff *Algorithmus* geht auf den persischen Gelehrten Muhammed Al Chwarizmi (8./9. Jhd.) zurück. Der älteste bekannte nicht-triviale Algorithmus ist der nach *Euklid* benannte Algorithmus zur Berechnung des größten gemeinsamen Teilers zweier natürlicher Zahlen (300 v. Chr.). Von einem Algorithmus wird erwartet, dass er jede *Problemeingabe* nach endlich vielen Rechenschritten löst (etwa durch Produktion einer Ausgabe). Ein Algorithmus ist ein „Verfahren“ zur Lösung eines Berechnungsproblems, das sich prinzipiell auf einer Turingmaschine implementieren lässt (Church-Turing-These).

Wir betrachten zunächst nur Entscheidungsprobleme, was der Berechnung von $\{0,1\}$-wertigen Funktionen entspricht. Problemeingaben können Zahlen, Formeln, Graphen etc. sein. Diese werden über einem *Eingabealphabet* Σ kodiert.

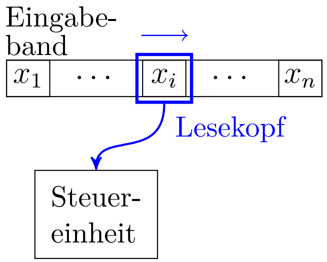
Definition 1. *Ein* ***Alphabet*** *ist eine geordnete endliche Menge* $\Sigma = \{a_1, \ldots, a_m\}$, $m \geq 1$, *von* ***Zeichen****. Eine Folge* $x = x_1 \ldots x_n \in \Sigma^n$ *heißt* ***Wort*** *(der* ***Länge*** n*). Die Menge aller Wörter über* Σ *ist*

$$\Sigma^* = \bigcup_{n\geq 0} \Sigma^n = \{x_1 \cdots x_n \mid n \geq 0 \text{ und } x_i \in \Sigma \text{ für } i = 1, \ldots, n\}.$$

Das (einzige) Wort der Länge $n = 0$ *ist das* ***leere Wort****, welches wir mit* ε *bezeichnen. Jede Teilmenge* $L \subseteq \Sigma^*$ *heißt* ***Sprache*** *über dem Alphabet* Σ.

2 Reguläre Sprachen

2.1 Endliche Automaten



Ein endlicher Automat ist eine „abgespeckte“ Turingmaschine, die nur kostant viel Speicherplatz benötigt und bei Eingaben der Länge n nur n Rechenschritte ausführt. Um die gesamte Eingabe lesen zu können, muss der Automat also in jedem Schritt ein Zeichen der Eingabe verarbeiten.

Definition 2. *Ein* ***endlicher Automat*** *(kurz: DFA; deterministic finite automaton) wird durch ein 5-Tupel* $M = (Z, \Sigma, \delta, q_0, E)$ *beschrieben, wobei*

- $Z \neq \emptyset$ *eine endliche Menge von* ***Zuständen****,*
- Σ *das* ***Eingabealphabet****,*
- $\delta : Z \times \Sigma \to Z$ *die* ***Überführungsfunktion****,*
- $q_0 \in Z$ *der* ***Startzustand*** *und*
- $E \subseteq Z$ *die Menge der* ***Endzustände*** *ist.*

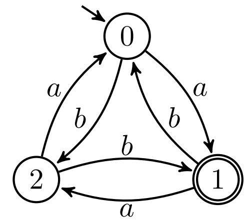
Die von M ***akzeptierte*** *oder* ***erkannte Sprache*** *ist*

$$L(M) = \left\{ x_1 \cdots x_n \in \Sigma^* \;\middle|\; \begin{array}{c} \exists\, q_1, \ldots, q_{n-1} \in Z, q_n \in E: \\ \delta(q_i, x_{i+1}) = q_{i+1} \text{ für } i = 0, \ldots, n-1 \end{array} \right\}.$$

Beispiel 3. *Betrachte den DFA* $M = (Z, \Sigma, \delta, 0, E)$ *mit* $Z = \{0, 1, 2\}$, $\Sigma = \{a, b\}$, $E = \{1\}$ *und der Überführungsfunktion*

δ	0	1	2
a	1	2	0
b	2	0	1

Graphische Darstellung:



Der Startzustand wird meist durch einen Pfeil und Endzustände werden durch einen doppelten Kreis gekennzeichnet. ◁

Bezeichne $\hat{\delta}(q, x)$ denjenigen Zustand, in dem sich M nach Lesen von x befindet, wenn M im Zustand q gestartet wird. Dann können wir die Funktion

$$\hat{\delta} : Z \times \Sigma^* \to Z$$

induktiv wie folgt definieren. Für $q \in Z$, $x \in \Sigma^*$ und $a \in \Sigma$ sei

$$\begin{aligned} \hat{\delta}(q, \varepsilon) &= q, \\ \hat{\delta}(q, xa) &= \delta(\hat{\delta}(q, x), a). \end{aligned}$$

Die von M erkannte Sprache lässt sich nun auch in der Form

$$L(M) = \{x \in \Sigma^* \mid \hat{\delta}(q_0, x) \in E\}$$

schreiben.

Behauptung 1. *Der DFA* M *aus Beispiel 3 akzeptiert die Sprache*

$$L(M) = \{x \in \Sigma^* \mid \#_a(x) - \#_b(x) \equiv 1 \pmod 3\},$$

wobei $\#_a(x)$ *die Anzahl der Vorkommen des Buchstabens a in x bezeichnet und* $j \equiv k \pmod m$ *bedeutet, dass* $j - k$ *durch* m *teilbar ist. Für* $j \equiv k \pmod m$ *schreiben wir im Folgenden auch kurz* $j \equiv_m k$.

Beweis. Da M nur den Endzustand 1 hat, ist $L(M) = \{x \in \Sigma^* \mid \hat{\delta}(0, x) = 1\}$. Daher reicht es, folgende Kongruenzgleichung zu zeigen:

$$\hat{\delta}(0, x) \equiv_3 \#_a(x) - \#_b(x).$$

Wir beweisen die Kongruenz induktiv über die Länge n von x.

Induktionsanfang ($n = 0$): klar, da $\hat{\delta}(0, \varepsilon) = \#_a(\varepsilon) = \#_b(\varepsilon) = 0$ ist.

Induktionsschritt ($n \rightsquigarrow n + 1$): Sei $x = x_1 \cdots x_{n+1}$ gegeben und sei $i = \hat{\delta}(0, x_1 \cdots x_n)$. Nach IV ist

$$i \equiv_3 \#_a(x_1 \cdots x_n) - \#_b(x_1 \cdots x_n).$$

Wegen $\delta(i, a) \equiv_3 i + 1$ und $\delta(i, b) \equiv_3 i - 1$ folgt

$$\delta(i, x_{n+1}) \equiv_3 i + \#_a(x_{n+1}) - \#_b(x_{n+1}) = \#_a(x) - \#_b(x).$$

Folglich ist

$$\hat{\delta}(0, x) = \delta(\hat{\delta}(0, x_1 \cdots x_n), x_{n+1}) = \delta(i, x_{n+1}) \equiv_3 \#_a(x) - \#_b(x).$$

■

Eine von einem DFA akzeptierte Sprache wird als **regulär** bezeichnet. Die zugehörige Sprachklasse ist

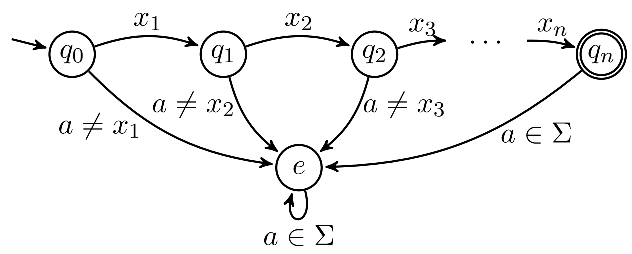
$$\mathsf{REG} = \{L(M) \mid M \text{ ist ein DFA}\}.$$

Um ein intuitives Verständnis für die Berechnungskraft von DFAs zu entwickeln, werden wir Antworten auf folgende Frage suchen.

Frage: Welche Sprachen gehören zu REG und welche nicht?

Dabei legen wir unseren Überlegungen ein beliebiges aber fest gewähltes Alphabet $\Sigma = \{a_1, \ldots, a_m\}$ zugrunde.

Beobachtung 4. *Alle Sprachen, die aus einem einzigen Wort $x = x_1 \cdots x_n \in \Sigma^*$ bestehen (diese Sprachen werden auch als* Singletonsprachen *bezeichnet), sind regulär. Für folgenden DFA M gilt nämlich $L(M) = \{x\}$.*



Formal lässt sich M also durch das Tupel $M = (Z, \Sigma, \delta, q_0, E)$ mit $Z = \{q_0, \ldots, q_n, e\}$, $E = \{q_n\}$ und der Überführungsfunktion

$$\delta(q, a_j) = \begin{cases} q_{i+1}, & q = q_i \text{ für ein } i \text{ mit } 0 \le i \le n-1 \text{ und } a_j = x_{i+1} \\ e, & \text{sonst} \end{cases}$$

beschreiben.

Als nächstes betrachten wir Abschlusseigenschaften der Sprachklasse REG.

Definition 5. *Ein* **(k-stelliger) Sprachoperator** *ist eine Abbildung op, die k Sprachen $L_1, \ldots, L_k$ auf eine Sprache op$(L_1, \ldots, L_k)$ abbildet.*

Beispiel 6. *Der 2-stellige Schnittoperator bildet zwei Sprachen L_1 und L_2 auf die Sprache $L_1 \cap L_2$ ab.* ◁

Definition 7. *Eine Sprachklasse $\mathcal{K}$ heißt unter op* **abgeschlossen***, wenn gilt:*

$$L_1, \ldots, L_k \in \mathcal{K} \Rightarrow op(L_1, \ldots, L_k) \in \mathcal{K}.$$

Der **Abschluss** *von $\mathcal{K}$ unter op ist die kleinste Sprachklasse $\mathcal{K}'$, die $\mathcal{K}$ enthält und unter op abgeschlossen ist.*

Definition 8. *Für eine Sprachklasse $\mathcal{C}$ bezeichne co-$\mathcal{C}$ die Klasse $\{\bar{L} \mid L \in \mathcal{C}\}$ aller Komplemente von Sprachen in $\mathcal{C}$.*

Es ist leicht zu sehen, dass $\mathcal{C}$ genau dann unter Komplementbildung abgeschlossen ist, wenn *co*-$\mathcal{C} = \mathcal{C}$ ist.

Beobachtung 9. *Mit* $L_1, L_2 \in \mathsf{REG}$ *sind auch die Sprachen* $\overline{L_1} = \Sigma^* \setminus L_1$, $L_1 \cap L_2$ *und* $L_1 \cup L_2$ *regulär. Sind nämlich* $M_i = (Z_i, \Sigma, \delta_i, q_0, E_i)$, $i = 1, 2$, *DFAs mit* $L(M_i) = L_i$, *so akzeptiert der DFA*

$$\overline{M_1} = (Z_1, \Sigma, \delta_1, q_0, Z_1 \setminus E_1)$$

das Komplement $\overline{L_1}$ *von* L_1. *Der Schnitt* $L_1 \cap L_2$ *von* L_1 *und* L_2 *wird dagegen von dem DFA*

$$M = (Z_1 \times Z_2, \Sigma, \delta, (q_0, q_0), E_1 \times E_2)$$

mit

$$\delta((q, p), a) = (\delta_1(q, a), \delta_2(p, a))$$

akzeptiert (M wird auch **Kreuzproduktautomat** *genannt). Wegen* $L_1 \cup L_2 = \overline{(\overline{L_1} \cap \overline{L_2})}$ *ist dann aber auch die Vereinigung von* L_1 *und* L_2 *regulär. (Wie sieht der zugehörige DFA aus?)*

Aus Beobachtung 9 folgt, dass alle endlichen und alle co-endlichen Sprachen regulär sind. Da die in Beispiel 3 betrachtete Sprache weder endlich noch co-endlich ist, haben wir damit allerdings noch nicht alle regulären Sprachen erfasst.

Es stellt sich die Frage, ob REG neben den mengentheoretischen Operationen Schnitt, Vereinigung und Komplement unter weiteren Operationen wie etwa der **Produktbildung**

$$L_1 L_2 = \{xy \mid x \in L_1, y \in L_2\}$$

(auch **Verkettung** oder **Konkatenation** genannt) oder der Bildung der **Sternhülle**

$$L^* = \bigcup_{n \geq 0} L^n$$

abgeschlossen ist. Die n-fache Potenz L^n von L ist dabei induktiv definiert durch

$$L^0 = \{\varepsilon\},\ L^{n+1} = L^n L.$$

Die **Plushülle** von L ist

$$L^+ = \bigcup_{n \geq 1} L^n = LL^*.$$

Ist $L_1 = \{x\}$ eine Singletonsprache, so schreiben wir für das Produkt $\{x\}L_2$ auch einfach xL_2.

Im übernächsten Abschnitt werden wir sehen, dass die Klasse REG als der Abschluss der endlichen Sprachen unter Vereinigung, Produktbildung und Sternhülle charakterisierbar ist.

Beim Versuch, einen endlichen Automaten für das Produkt $L_1 L_2$ zweier regulärer Sprachen zu konstruieren, stößt man auf die Schwierigkeit, den richtigen Zeitpunkt für den Übergang von (der Simulation von) M_1 zu M_2 zu finden. Unter Verwendung eines nichtdeterministischen Automaten lässt sich dieses Problem jedoch leicht beheben, da dieser den richtigen Zeitpunkt „erraten“ kann.

Im nächsten Abschnitt werden wir nachweisen, dass auch nichtdeterministische endliche Automaten nur reguläre Sprachen erkennen können.

2.2 Nichtdeterministische endliche Automaten

Definition 10. *Ein* ***nichtdeterministischer endlicher Automat*** *(kurz: NFA; nondeterministic finite automaton)* $N = (Z, \Sigma, \delta, Q_0, E)$ *ist ähnlich aufgebaut wie ein DFA, nur dass er mehrere Startzustände (zusammengefasst in der Menge* $Q_0 \subseteq Z$*) haben kann und seine Überführungsfunktion die Form*

$$\delta : Z \times \Sigma \to \mathcal{P}(Z)$$

hat. Hierbei bezeichnet $\mathcal{P}(Z)$ die Potenzmenge *(also die Menge aller Teilmengen) von Z. Diese wird auch oft mit 2^Z bezeichnet. Die von N akzeptierte Sprache ist*

$$L(N) = \left\{ x_1 \cdots x_n \in \Sigma^* \;\middle|\; \begin{array}{c} \exists\, q_0 \in Q_0, q_1, \ldots, q_{n-1} \in Z, q_n \in E\colon \\ q_{i+1} \in \delta(q_i, x_{i+1}) \textit{ für } i = 0, \ldots, n-1 \end{array} \right\}.$$

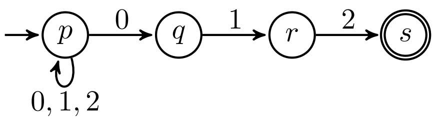
Ein NFA kann also nicht nur eine, sondern mehrere verschiedene Rechnungen ausführen. Die Eingabe gehört bereits dann zu $L(N)$, wenn bei einer dieser Rechnungen nach Lesen des gesamten Eingabewortes ein Endzustand erreicht wird.

Im Gegensatz zu einem DFA, dessen Überführungsfunktion auf der gesamten Menge $Z \times \Sigma$ definiert ist, kann ein NFA „stecken bleiben". Das ist dann der Fall, wenn er in einen Zustand q gelangt, in dem das nächste Eingabezeichen x_i wegen $\delta(q, x_i) = \emptyset$ nicht gelesen werden kann.

Beispiel 11. *Betrachte den NFA $N = (Z, \Sigma, \delta, Q_0, E)$ mit Zustandsmenge $Z = \{p, q, r, s\}$, Eingabealphabet $\Sigma = \{0, 1, 2\}$, Start- und Endzustandsmenge $Q_0 = \{p\}$ und $E = \{s\}$ sowie der Überführungsfunktion*

δ	p	q	r	s
0	$\{p, q\}$	$\emptyset$	$\emptyset$	$\emptyset$
1	$\{p\}$	$\{r\}$	$\emptyset$	$\emptyset$
2	$\{p\}$	$\emptyset$	$\{s\}$	$\emptyset$

Graphische Darstellung:



Offensichtlich akzeptiert N die Sprache $L(N) = \{x012 \mid x \in \Sigma^*\}$ aller Wörter, die mit dem Suffix 012 enden. ◁

Beobachtung 12. *Sind $N_i = (Z_i, \Sigma, \delta_i, Q_i, E_i)$ $(i = 1, 2)$ NFAs, so werden auch die Sprachen $L(N_1)L(N_2)$ und $L(N_1)^*$ von einem NFA erkannt. Wir können $Z_1 \cap Z_2 = \emptyset$ annehmen. Dann akzeptiert der NFA*

$$N = (Z_1 \cup Z_2, \Sigma, \delta, Q_1, E)$$

mit

$$\delta(p, a) = \begin{cases} \delta_1(p, a), & p \in Z_1 \setminus E_1, \\ \delta_1(p, a) \cup \bigcup_{q \in Q_2} \delta_2(q, a), & p \in E_1, \\ \delta_2(p, a), & \textit{sonst} \end{cases}$$

und

$$E = \begin{cases} E_1 \cup E_2, & Q_2 \cap E_2 \neq \emptyset \\ E_2, & \textit{sonst} \end{cases}$$

die Sprache $L(N_1)L(N_2)$ und der NFA

$$N^* = (Z_1 \cup \{q_{neu}\}, \Sigma, \delta^*, Q_1 \cup \{q_{neu}\}, E_1 \cup \{q_{neu}\})$$

mit

$$\delta^*(p, a) = \begin{cases} \delta(p, a) \cup \bigcup_{q \in Q_1} \delta(q, a), & p \in E_1, \\ \delta(p, a), & \textit{sonst} \end{cases}$$

die Sprache $L(N_1)^*$.

Satz 13 (Rabin und Scott)**.**
$\mathsf{REG} = \{L(N) \mid N \textit{ ist ein NFA}\}$.

Beweis. Die Inklusion von links nach rechts ist klar, da jeder DFA auch als NFA aufgefasst werden kann. Für die Gegenrichtung konstruieren wir zu einem NFA $N = (Z, \Sigma, \delta, Q_0, E)$ einen DFA $M = (\mathcal{P}(Z), \Sigma, \delta', Q_0, E')$ mit $L(M) = L(N)$. Wir definieren die Überführungsfunktion $\delta' : \mathcal{P}(Z) \times \Sigma \to \mathcal{P}(Z)$ von M mittels

$$\delta'(Q, a) = \bigcup_{q \in Q} \delta(q, a).$$

Die Menge $\delta'(Q, a)$ enthält also alle Zustände, in die N gelangen kann, wenn N ausgehend von einem beliebigen Zustand $q \in Q$ das Zeichen a liest. Intuitiv bedeutet dies, dass der DFA M den NFA N simuliert, indem M in seinem aktuellen Zustand Q die Information speichert, in welchen Zuständen sich N momentan befinden könnte. Für die Erweiterung $\hat{\delta}' : \mathcal{P}(Z) \times \Sigma^* \to \mathcal{P}(Z)$ von δ' (siehe Seite 2) können wir nun folgende Behauptung zeigen:

$\hat{\delta}'(Q_0, x)$ enthält alle Zustände, die N ausgehend von einem Startzustand nach Lesen der Eingabe x erreichen kann.

Wir beweisen die Behauptung induktiv über die Länge n von x.

Induktionsanfang ($n = 0$): klar, da $\hat{\delta}'(Q_0, \varepsilon) = Q_0$ ist.

Induktionsschritt ($n - 1 \rightsquigarrow n$): Sei $x = x_1 \cdots x_n$ gegeben. Nach Induktionsvoraussetzung enthält

$$Q_{n-1} = \hat{\delta}'(Q_0, x_1 \cdots x_{n-1})$$

alle Zustände, die $N(x)$ in genau $n - 1$ Schritten erreichen kann. Wegen

$$\hat{\delta}'(Q_0, x) = \delta'(Q_{n-1}, x_n) = \bigcup_{q \in Q_{n-1}} \delta(q, x_n)$$

enthält dann aber $\hat{\delta}'(Q_0, x)$ alle Zustände, die $N(x)$ in genau n Schritten erreichen kann.

Deklarieren wir nun diejenigen Teilmengen $Q \subseteq Z$, die mindestens einen Endzustand von N enthalten, als Endzustände des **Potenzmengenautomaten** M, d.h.

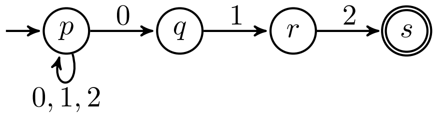
$$E' = \{Q \subseteq Z \mid Q \cap E \neq \emptyset\},$$

so folgt für alle Wörter $x \in \Sigma^*$:

$$\begin{aligned} x \in L(N) \quad &\Leftrightarrow \quad N(x) \text{ kann in genau } |x| \text{ Schritten einen Endzustand erreichen} \\ &\Leftrightarrow \quad \hat{\delta}'(Q_0, x) \cap E \neq \emptyset \\ &\Leftrightarrow \quad \hat{\delta}'(Q_0, x) \in E' \\ &\Leftrightarrow \quad x \in L(M). \end{aligned}$$

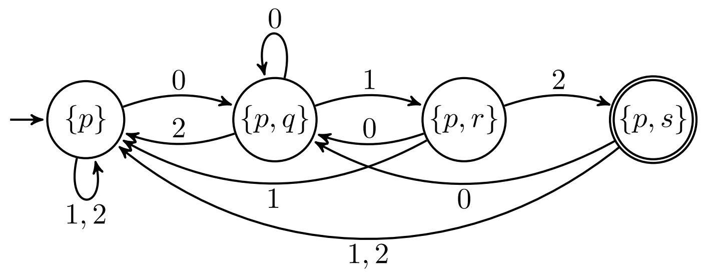
■

Beispiel 14. *Für den NFA $N = (Z, \Sigma, \delta, Q_0, E)$ aus Beispiel 11*



ergibt die Konstruktion des vorigen Satzes den folgenden DFA M (nach Entfernen aller vom Startzustand $Q_0 = \{p\}$ aus nicht erreichbaren Zustände):

δ'	0	1	2
$Q_0 = \{p\}$	$\{p, q\}$	$\{p\}$	$\{p\}$
$Q_1 = \{p, q\}$	$\{p, q\}$	$\{p, r\}$	$\{p\}$
$Q_2 = \{p, r\}$	$\{p, q\}$	$\{p\}$	$\{p, s\}$
$Q_3 = \{p, s\}$	$\{p, q\}$	$\{p\}$	$\{p\}$



◁

Im obigen Beispiel wurden für die Konstruktion des DFA M aus dem NFA N nur 4 der insgesamt $2^{\|Z\|} = 16$ Zustände benötigt, da die übrigen 12 Zustände in $\mathcal{P}(Z)$ nicht vom Startzustand $Q_0 = \{p\}$ aus erreichbar sind. Es gibt jedoch Beispiele, bei denen alle $2^{\|Z\|}$ Zustände in $\mathcal{P}(Z)$ für die Konstruktion des Potenzmengenautomaten benötigt werden (siehe Übungen).

Korollar 15. *Die Klasse* REG *der regulären Sprachen ist unter folgenden Operationen abgeschlossen:*

- *Komplement,*
- *Durchschnitt,*
- *Vereinigung,*
- *Produkt,*
- *Sternhülle.*

2.3 Reguläre Ausdrücke

Wir haben uns im letzten Abschnitt davon überzeugt, dass auch NFAs nur reguläre Sprachen erkennen können:

$$\mathsf{REG} = \{L(M) \mid M \text{ ist ein DFA}\} = \{L(N) \mid N \text{ ist ein NFA}\}.$$

In diesem Abschnitt werden wir eine weitere Charakterisierung der regulären Sprachen kennen lernen:

> REG ist die Klasse aller Sprachen, die sich mittels der Operationen Vereinigung, Durchschnitt, Komplement, Produkt und Sternhülle aus der leeren Menge und den Singletonsprachen bilden lassen.

Tatsächlich kann hierbei sogar auf die Durchschnitts- und Komplementbildung verzichtet werden.

Definition 16. *Die Menge der* ***regulären Ausdrücke*** γ *(über einem Alphabet Σ) und die durch γ dargestellte Sprache $L(\gamma)$ sind induktiv wie folgt definiert. Die Symbole $\emptyset$, ϵ und a ($a \in \Sigma$) sind reguläre Ausdrücke, die*

- *die leere Sprache $L(\emptyset) = \emptyset$,*
- *die Sprache $L(\epsilon) = \{\varepsilon\}$ und*
- *für jedes Zeichen $a \in \Sigma$ die Sprache $L(a) = \{a\}$*

beschreiben. Sind α und β reguläre Ausdrücke, die die Sprachen $L(\alpha)$ und $L(\beta)$ beschreiben, so sind auch $\alpha\beta$, $(\alpha|\beta)$ und $(\alpha)^*$ reguläre Ausdrücke, die die Sprachen

- *$L(\alpha\beta) = L(\alpha)L(\beta)$,*
- *$L(\alpha|\beta) = L(\alpha) \cup L(\beta)$ und*
- *$L((\alpha)^*) = L(\alpha)^*$*

beschreiben.

Beispiel 17. *Die regulären Ausdrücke ϵ^*, $\emptyset^*$, $(0|1)^*00$ und $(\epsilon 0|\emptyset 1^*)$ beschreiben folgende Sprachen:*

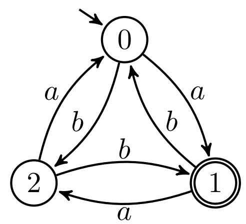
γ	ϵ^*	$\emptyset^*$	$(0\|1)^*00$	$(\epsilon 0\|\emptyset 1^*)$
$L(\gamma)$	$\{\varepsilon\}^* = \{\varepsilon\}$	$\emptyset^* = \{\varepsilon\}$	$\{x00 \mid x \in \{0,1\}^*\}$	$\{0\}$

◁

Bemerkung 18.

- *Um Klammern zu sparen, definieren wir folgende* ***Präzedenzordnung****: Der Sternoperator * bindet stärker als der Produktoperator und dieser wiederum stärker als der Vereinigungsoperator. Für $((a|b(c)^*)|d)$ können wir also kurz $a|bc^*|d$ schreiben.*
- *Da der reguläre Ausdruck $\gamma\gamma^*$ die Sprache $L(\gamma)^+$ beschreibt, verwenden wir γ^+ als Abkürzung für den Ausdruck $\gamma\gamma^*$.*

Beispiel 19. *Betrachte nebenstehenden DFA M. Um für die von M erkannte Sprache*



$$L(M) = \{x \in \{a,b\}^* \mid \#_a(x) - \#_b(x) \equiv_3 1\}$$

einen regulären Ausdruck zu finden, betrachten wir zunächst die Sprache

$$L_0 = \{x \in \{a,b\}^* \mid \#_a(x) - \#_b(x) \equiv_3 0\}.$$

L_0 enthält also alle Wörter x, die den DFA M ausgehend vom Zustand 0 in den Zustand 0 überführen. Jedes solche x setzt sich aus beliebig vielen Teilwörtern y zusammen, die M vom Zustand 0 in den Zustand 0 überführen, ohne zwischendurch den Zustand 0 anzunehmen. Jedes solche y beginnt entweder mit einem a (Übergang von 0 nach 1) oder mit einem b (Übergang von 0 nach 2). Im ersten Fall folgt eine beliebige Anzahl von Teilwörtern ab (Wechsel zwischen 1 und 2), an die sich entweder das Suffix aa (Rückkehr von 1 nach 0 über 2) oder das Suffix b (direkte Rückkehr von 1 nach 0) anschließt.

Analog folgt im zweiten Fall eine beliebige Anzahl von Teilwörtern ba (Wechsel zwischen 2 *und* 1*), an die sich entweder das Suffix a (direkte Rückkehr von* 2 *nach* 0*) oder das Suffix bb (Rückkehr von* 2 *nach* 0 *über* 1*) anschließt. Daher lässt sich* L_0 *durch den regulären Ausdruck*

$$\gamma_0 = (a(ab)^*(aa|b) \mid b(ba)^*(a|bb))^*$$

beschreiben. Eine ähnliche Überlegung zeigt, dass sich die Wörter, die M *ausgehend von* 0 *in den Zustand* 1 *überführen, ohne dass zwischendurch der Zustand* 0 *nochmals besucht wird, durch den regulären Ausdruck* $(a|bb)(ab)^*$ *beschrieben werden. Somit erhalten wir für* $L(M)$ *den regulären Ausdruck* $\gamma = \gamma_0(a|bb)(ab)^*$. ◁

Satz 20. $\mathsf{REG} = \{L(\gamma) \mid \gamma$ *ist ein regulärer Ausdruck*$\}$.

Beweis. Die Inklusion von rechts nach links ist klar, da die Basisausdrücke $\emptyset$, ϵ und a, $a \in \Sigma^*$, nur reguläre Sprachen beschreiben und die Sprachklasse REG unter Produkt, Vereinigung und Sternhülle abgeschlossen ist (siehe Beobachtungen 9 und 12).

Für die Gegenrichtung konstruieren wir zu einem DFA M einen regulären Ausdruck γ mit $L(\gamma) = L(M)$. Sei also $M = (Z, \Sigma, \delta, q_0, E)$ ein DFA, wobei wir annehmen können, dass $Z = \{1, \ldots, m\}$ und $q_0 = 1$ ist. Dann lässt sich $L(M)$ als Vereinigung

$$L(M) = \bigcup_{q \in E} L_{1,q}$$

von Sprachen der Form

$$L_{p,q} = \{x \in \Sigma^* \mid \hat{\delta}(p, x) = q\}$$

darstellen. Folglich reicht es zu zeigen, dass die Sprachen $L_{p,q}$ durch reguläre Ausdrücke beschreibbar sind. Hierzu betrachten wir die Sprachen

$$L^r_{p,q} = \left\{ x_1 \cdots x_n \in \Sigma^* \;\middle|\; \begin{array}{c} \hat{\delta}(p, x_1 \cdots x_n) = q \text{ und für} \\ i = 1, \ldots, n-1 \text{ gilt } \hat{\delta}(p, x_1 \cdots x_i) \le r \end{array} \right\}.$$

Wegen $L_{p,q} = L^m_{p,q}$ reicht es, reguläre Ausdrücke $\gamma^r_{p,q}$ für die Sprachen $L^r_{p,q}$ anzugeben. Im Fall $r = 0$ enthält

$$L^0_{p,q} = \begin{cases} \{a \in \Sigma \mid \delta(p, a) = q\} \cup \{\varepsilon\}, & p = q, \\ \{a \in \Sigma \mid \delta(p, a) = q\}, & \text{sonst} \end{cases}$$

nur Buchstaben (und eventuell das leere Wort) und ist somit leicht durch einen regulären Ausdruck $\gamma^0_{p,q}$ beschreibbar. Wegen

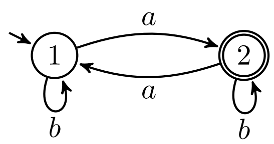
$$L^{r+1}_{p,q} = L^r_{p,q} \cup L^r_{p,r+1}(L^r_{r+1,r+1})^* L^r_{r+1,q}$$

lassen sich aus den regulären Ausdrücken $\gamma^r_{p,q}$ für die Sprachen $L^r_{p,q}$ leicht reguläre Ausdrücke für die Sprachen $L^{r+1}_{p,q}$ gewinnen:

$$\gamma^{r+1}_{p,q} = \gamma^r_{p,q} | \gamma^r_{p,r+1}(\gamma^r_{r+1,r+1})^* \gamma^r_{r+1,q}.$$

■

Beispiel 21. *Betrachte den DFA*



Da M *insgesamt* $m = 2$ *Zustände und nur den Endzustand* 2 *besitzt, ist*

$$L(M) = \bigcup_{q \in E} L_{1,q} = L_{1,2} = L^2_{1,2} = L(\gamma^2_{1,2}).$$

Um $\gamma^2_{1,2}$ *zu berechnen, benutzen wir die Rekursionsformel*

$$\gamma^{r+1}_{p,q} = \gamma^r_{p,q} | \gamma^r_{p,r+1}(\gamma^r_{r+1,r+1})^* \gamma^r_{r+1,q}$$

und erhalten

$$\begin{aligned} \gamma^2_{1,2} &= \gamma^1_{1,2} | \gamma^1_{1,2}(\gamma^1_{2,2})^* \gamma^1_{2,2}, \\ \gamma^1_{1,2} &= \gamma^0_{1,2} | \gamma^0_{1,1}(\gamma^0_{1,1})^* \gamma^0_{1,2}, \\ \gamma^1_{2,2} &= \gamma^0_{2,2} | \gamma^0_{2,1}(\gamma^0_{1,1})^* \gamma^0_{1,2}. \end{aligned}$$

Um den regulären Ausdruck $\gamma^2_{1,2}$ für $L(M)$ zu erhalten, genügt es also, die regulären Ausdrücke $\gamma^0_{1,1}$, $\gamma^0_{1,2}$, $\gamma^0_{2,1}$, $\gamma^0_{2,2}$, $\gamma^1_{1,2}$ und $\gamma^1_{2,2}$ zu berechnen:

r	p,q			
	$1,1$	$1,2$	$2,1$	$2,2$
0	$\epsilon\mid b$	a	a	$\epsilon\mid b$
1	-	$\underbrace{a\mid(\epsilon\mid b)(\epsilon\mid b)^*a}_{b^*a}$	-	$\underbrace{(\epsilon\mid b)\mid a(\epsilon\mid b)^*a}_{\epsilon\mid b\mid ab^*a}$
2	-	$\underbrace{b^*a\mid b^*a(\epsilon\mid b\mid ab^*a)^*(\epsilon\mid b\mid ab^*a)}_{b^*a(b\mid ab^*a)^*}$	-	-

◁

Korollar 22. *Sei L eine Sprache. Dann sind folgende Aussagen äquivalent:*

- *L ist regulär,*
- *es gibt einen DFA M mit $L = L(M)$,*
- *es gibt einen NFA N mit $L = L(N)$,*
- *es gibt einen regulären Ausdruck γ mit $L = L(\gamma)$,*
- *L lässt sich mit den Operationen Vereinigung, Produkt und Sternhülle aus endlichen Sprachen gewinnen,*
- *L lässt sich mit den Operationen $\cap$, $\cup$, Komplement, Produkt und Sternhülle aus endlichen Sprachen gewinnen.*

Wir werden bald noch eine weitere Charakterisierung von REG kennenlernen, nämlich durch reguläre Grammatiken. Zuvor befassen wir uns jedoch mit dem Problem, DFAs zu minimieren. Dabei spielen Relationen (insbesondere Äquivalenzrelationen) eine wichtige Rolle.

2.4 Relationalstrukturen

Sei A eine nichtleere Menge, R_i eine k_i-stellige Relation auf A, d.h. $R_i \subseteq A^{k_i}$ für $i = 1, \ldots, n$. Dann heißt $(A; R_1, \ldots, R_n)$ ***Relationalstruktur***. Die Menge A heißt ***Grundmenge***, ***Trägermenge*** oder ***Individuenbereich*** der Relationalstruktur.

Wir werden hier hauptsächlich den Fall $n = 1$, $k_1 = 2$, also (A, R) mit $R \subseteq A \times A$ betrachten. Man nennt dann R eine **(binäre) Relation** auf A. Oft wird für $(a, b) \in R$ auch die **Infix-Schreibweise** aRb benutzt.

Beispiel 23.

- *(F, M) mit $F = \{f \mid f$ ist Fluss in Europa$\}$ und*

$$M = \{(f, g) \in F \times F \mid f \text{ mündet in } g\}.$$

- *(U, B) mit $U = \{x \mid x$ ist Berliner$\}$ und*

$$B = \{(x, y) \in U \times U \mid x \text{ ist Bruder von } y\}.$$

- *$(\mathsf{P}(M), \subseteq)$, wobei $\mathsf{P}(M)$ die Potenzmenge einer beliebigen Menge M und $\subseteq$ die Inklusionsbeziehung auf den Teilmengen von M ist.*
- *(A, Id_A), wobei $Id_A = \{(x, x) \mid x \in A\}$ die **Identität auf** A ist.*
- *$(\mathbb{R}, \leq)$.*
- *$(\mathbb{Z}, |)$, wobei $|$ die "teilt"-Relation bezeichnet.*
- *$(\mathcal{F}ml, \Rightarrow)$ mit $\mathcal{F}ml = \{F \mid F$ ist aussagenlogische Formel$\}$ und*

$$\Rightarrow\ =\ \{(F, G) \in \mathcal{F}ml \times \mathcal{F}ml \mid G \text{ ist Folgerung von } F\}. \quad \triangleleft$$

Da Relationen Mengen sind, sind auf ihnen die mengentheoretischen Operationen ***Durchschnitt***, ***Vereinigung***, ***Komplement*** und ***Differenz*** definiert. Seien R und S Relationen auf A, dann ist

$$\begin{aligned} R \cap S &= \{(x,y) \in A \times A \mid xRy \wedge xSy\}, \\ R \cup S &= \{(x,y) \in A \times A \mid xRy \vee xSy\}, \\ R - S &= \{(x,y) \in A \times A \mid xRy \wedge \neg xSy\}, \\ \overline{R} &= (A \times A) - R. \end{aligned}$$

Sei allgemeiner $\mathcal{M} \subseteq \mathcal{P}(A \times A)$ eine beliebige Menge von Relationen auf A. Dann sind der **Schnitt über** $\mathcal{M}$ und die **Vereinigung über** $\mathcal{M}$ folgende Relationen:

$$\begin{aligned} \bigcap \mathcal{M} &= \{(x,y) \mid \forall R \in \mathcal{M} : xRy\}, \\ \bigcup \mathcal{M} &= \{(x,y) \mid \exists R \in \mathcal{M} : xRy\}. \end{aligned}$$

Weiterhin ist die **Inklusionsrelation** $R \subseteq S$ auf Relationen von Bedeutung:

$$R \subseteq S \Leftrightarrow \forall x, y : xRy \rightarrow xSy.$$

Die ***transponierte (konverse) Relation*** zu R ist

$$R^T = \{(y,x) \mid xRy\}.$$

R^T wird oft auch mit R^{-1} bezeichnet. Zum Beispiel ist $(\mathbb{R}, \leq^T) = (\mathbb{R}, \geq)$.

Seien R und S Relationen auf A. Das ***Produkt*** oder die ***Komposition*** von R und S ist

$$R \circ S = \{(x,z) \in A \times A \mid \exists y \in A : xRy \wedge ySz\}.$$

Beispiel 24. *Ist B die Relation "ist Bruder von", V "ist Vater von", M "ist Mutter von" und $E = V \cup M$ "ist Elternteil von", so ist $B \circ E$ die Onkel-Relation.* ◁

Übliche Bezeichnungen für das Relationenprodukt sind auch $R;S$ und $R \cdot S$ oder einfach RS. Das n-fache Relationenprodukt $R \circ \cdots \circ R$ von R wird mit R^n bezeichnet. Dabei ist $R^0 = Id$.

Vorsicht: Das n-fache Relationenprodukt R^n von R sollte nicht mit dem n-fachen kartesischen Produkt $R \times \cdots \times R$ der *Menge* R verwechselt werden. Wir vereinbaren, dass R^n das n-fache Relationenprodukt bezeichnen soll, falls R eine Relation ist.

Eigenschaften von Relationen

Sei R eine Relation auf A. Dann heißt R

reflexiv,	falls $\forall x \in A : xRx$	(also $Id_A \subseteq R$)
irreflexiv,	falls $\forall x \in A : \neg xRx$	(also $Id_A \subseteq \overline{R}$)
symmetrisch,	falls $\forall x, y \in A : xRy \Rightarrow yRx$	(also $R \subseteq R^T$)
asymmetrisch,	falls $\forall x, y \in A : xRy \Rightarrow \neg yRx$	(also $R \subseteq \overline{R^T}$)
antisymmetrisch,	falls $\forall x, y \in A : xRy \wedge yRx \Rightarrow x = y$	(also $R \cap R^T \subseteq Id$)
konnex,	falls $\forall x, y \in A : xRy \vee yRx$	(also $A \times A \subseteq R \cup R^T$)
semikonnex,	falls $\forall x, y \in A : x \neq y \Rightarrow xRy \vee yRx$	(also $\overline{Id} \subseteq R \cup R^T$)
transitiv,	falls $\forall x, y, z \in A : xRy \wedge yRz \Rightarrow xRz$	(also $R^2 \subseteq R$)

gilt.

Beispiel 25.

- *Die Relation "ist Schwester von" ist zwar in einer reinen Damengesellschaft symmetrisch, i.a. jedoch weder symmetrisch noch asymmetrisch noch antisymmetrisch.*
- $(\mathbb{R}, <)$ *ist irreflexiv, asymmetrisch, transitiv und semikonnex.*
- $(\mathbb{R}, \leq)$ *und* $(\mathsf{P}(M), \subseteq)$ *sind reflexiv, antisymmetrisch und transitiv.*
- $(\mathbb{R}, \leq)$ *ist auch konnex und* $(\mathsf{P}(M), \subseteq)$ *ist im Fall* $\|M\| \leq 1$ *zwar auch konnex, aber im Fall* $\|M\| \geq 2$ *weder semikonnex noch konnex.* ◁